Discounted Cuts: A Stackelberg Approach to Network Disruption

Pål Grønås Drange Fedor V. Fomin Petr Golovach
Danil Sagunov*

November 17, 2025

Abstract

We study a Stackelberg variant of the classical Most Vital Links problem, modeled as a one-round adversarial game between an attacker and a defender. The attacker strategically removes up to k edges from a flow network to maximally disrupt flow between a source s and a sink t, after which the defender optimally reroutes the remaining flow. To capture this attacker–defender interaction, we introduce a new mathematical model of *discounted cuts*, in which the cost of a cut is evaluated by excluding its k most expensive edges. This model generalizes the Most Vital Links problem and uncovers novel algorithmic and complexity-theoretic properties.

We develop a unified algorithmic framework for analyzing various forms of discounted cut problems, including minimizing or maximizing the cost of a cut under discount mechanisms that exclude either the k most expensive or the k cheapest edges. While most variants are NP-complete on general graphs, our main result establishes polynomial-time solvability for all discounted cut problems in our framework when the input is restricted to bounded-genus graphs, a relevant class that includes many real-world networks such as transportation and infrastructure networks. With this work, we aim to open collaborative bridges between artificial intelligence, algorithmic game theory, and operations research.

1 Introduction

Consider the following scenario of securing a network: An attacker attempts to escalate privileges from server s to a critical database server t. The security team has a limited budget β to deploy firewalls or physically disconnect cables. Some links are very expensive to disable—typically those that are main trunk lines carrying heavy traffic. However, the company's cybersecurity insurance allows the team to discount the cost of disabling up to k links, either by covering the most expensive ones or, alternatively, the least expensive.

As another example, consider a multi-agent system where agents are connected via conflict relationships. We aim to divide the agents into two teams or coalitions in a way that maximizes inter-group tension. When agents are represented as

*This author was supported by the Ministry of Economic Development of the Russian Federation (IGK 000000C313925 P4C0002), agreement No. 139-15-2025-010.

nodes and interaction strengths as weighted edges, this leads to a natural MaxCut formulation: the goal is to partition the graph to maximize the total weight of edges crossing the cut. However, in many realistic scenarios, some interactions may be negligible or unreliable—such as low-trust links, wrong information, or edges that can be manipulated by a bounded-perception attacker. In such cases, we would be more interested in solving the *discounted maximum cut*, where the objective function ignores the k weakest (i.e., lowest-weight) edges in the cut and thus to focus on strong, strategically meaningful interactions, while discounting minor or noisy ones.

This gives rise to a Stackelberg variant of the classic cut problems, where a leader chooses a cut and a follower modifies the cost of up to k edges post hoc. Depending on whether the follower discounts the most or the least expensive edges, the effective cost of the cut—and thus the optimal strategy—changes significantly. See Figure 1 for an example.

Although the discount variants of cut problems appear closely related to the standard cut problems and to classical interdiction models such as the Most Vital Links problem (introduced by Wollmer [23, 24]), they give rise to novel algorithmic questions. In particular, the cost function becomes non-additive and adversarially dependent on the edge weights, breaking standard structural properties of cuts.

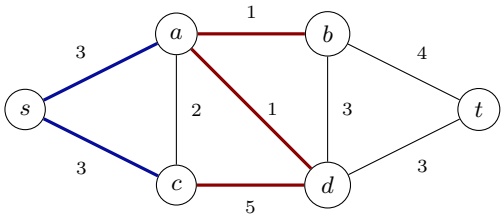

Figure 1: A minimum s-t-cut in this example is $\{sa, sc\}$ (blue) of cost 6. For $k = 1$, the minimum discounted s-t-cut is $\{ab, ad, cd\}$ (red), with discounted cost 2. This corresponds to the edge cd being the *most vital link*, i.e., the removal of cd leads to the largest decrease in maximum flow.

1.1 Cuts with k Free Edges

In this paper, we introduce and study several variants of the minimum and maximum cut problems in which the cost of some edges in the cut is discounted. Specifically, we consider two discount mechanisms: one in which we do not pay for the k most expensive edges, and another in which we do not pay for the k cheapest edges. These correspond to the two natural game strategies of always discounting the most expensive, and most cheap edges. We present these discounted-cost cut problems as formal models of network vulnerability and adversarial flow inhibition, providing a suite of complexity and algorithmic results with a particular focus on tractable cases in graphs of bounded genus. Our findings advance the algorithmic understanding of partial cuts under adversarial or cooperative discounting, opening new directions in network design and interdiction.

Let S be an ordered list of numbers, ordered from smallest to largest. We define two discounted cost functions

$$\mathsf{Cost}_k^{\text{exp}}(S) = \sum_{i}^{k} S_i \quad \text{and} \quad \mathsf{Cost}_k^{\text{cheap}}(S) = \sum_{n-k}^{n} S_i,$$

i.e., $\mathsf{Cost}_k^{\text{exp}}$ (resp. $\mathsf{Cost}_k^{\text{cheap}}$) is the sum of the elements in S *except* the k most (resp. least) expensive elements. In this work we will be talking mostly about cuts, so we define the following. Given a cut (A, B) of a graph G with an (edge) cost function $c\colon E(G) \to \mathbb{Z}_{\geq 0}$ and an integer $k \geq 0$, we define the *discounted cost* of the cut (A, B) as

$$\begin{aligned}\mathsf{Cost}_k^{\text{exp}}(A, B) &= c(E(A, B)) - c(R) \\ &= \sum_{e \in E(A,B)} c(e) - \sum_{e \in R} c(e),\end{aligned}$$

where R is the set of k most expensive edges of the edge cut set $E(A, B)$. (We assume $\mathsf{Cost}_k^{\text{exp}}(A, B) = 0$ whenever $|E(A, B)| \leq k$.) We consider the following variant of the classic MinCut problem Min s-t-Cut with k Free Expensive Edges: *Does there exist an s-t-cut (A, B) of G of discounted cost $\mathsf{Cost}_k^{\text{exp}}(A, B) \leq \beta$, given two terminal vertices s and t, an integer $k \geq 0$, and the budget $\beta \geq 0$.* Let us note that for $k = 0$, Min s-t-Cut$-k$ exp is the problem of finding a minimum s-t-cut.

A close well-studied relative of the minimum s-t-cut problem is the Global Min-cut, or the edge connectivity of a graph. Here the task is to identify the set of edges of minimum weight in a connected graph whose removal disconnects the graph. We define the discounted version of Global Min-cut as follows, MinCut with k Free Expensive Edges: Does there exist a cut (A, B) of G such that $\mathsf{Cost}_k^{\text{exp}}(A, B) \leq \beta$, given an integer $k \geq 0$, and the budget $\beta \geq 0$. The problem MinCut with k Free Cheap Edges, or MinCut$-k$ cheap for short, is defined similarly.

Another important cut problem is the Maximum Cut problem, where the goal is to partition a graph to maximize the total weight of edges crossing the cut. This problem is significantly more challenging than the corresponding minimization variants. Analogously, we define four versions of the Maximum Cut problem based on the presence or absence of terminals s and t, and the choice of discount function (excluding either cheap or expensive edges). In total, we thus obtain eight distinct discounted cut problems (see Table 1).

1.2 Our Results and Methods

In this section, we provide an overview of our main results. A central theme in our work is the use of a series of reductions that transform various discounted cut problems into their classic counterparts, enabling the application of classical techniques in new, cost-aware settings. Building on this, we provide complexity classification for all variants of the discounted cut problem. Together, these results offer a comprehensive view of the computational landscape for discounted cuts and reveal several surprising gaps in complexity between closely related problems. We summarize our results in Table 1.

	s-t-cut		Global cut	
	General	Planar	General	Planar
Min Exp.	W[1]-hard (Thm 1)	P (Thm 4)	P (Thm 5)	P (Thm 5)
Max Cheap	paraNP-hard	P *(i)* (Thm 2)	paraNP-hard	P *(i)* (Thm 2)
Min Cheap	P (Theorem 3)	P (Thm 3)	P (Thm 3)	P (Thm 3)
Max Exp.	paraNP-hard	P *(i)* (Thm 2)	paraNP-hard	P (Thm 3)

Table 1: Complexity classification of eight discounted-cut problems (minimum/-maximum cut, s-t/global cut, k cheap/k expensive discounting), each studied on general and planar (bounded-genus) graphs, resulting in 16 complexity entries. The complexity is parameterized by k; entries marked "(i)" indicate tractability results known only for polynomial costs.

Surprisingly, despite the long history of Min s-t-Cut$-k$ exp[1], we have not found an NP-hardness proof in the literature. The only known complexity lower bound is an NP-hardness result for a more general *network interdiction* model, in which each edge e has a resource cost $r(e)$. The goal in this model is to reduce the capacity of every s-t-cut by deleting a set of edges whose total resource cost does not exceed a threshold R. In this setting, Min s-t-Cut$-k$ exp is a special case where $r(e) = 1$ for every edge of G and $R = k$. A simple reduction from Knapsack shows that this more general interdiction problem is *weakly* NP-complete [3, 25]. Our first result significantly strengthens the known intractability bounds by establishing NP-completeness of Min s-t-Cut$-k$ exp when edge costs are restricted to one and two.

Theorem 1. Min s-t-Cut$-k$ exp *is NP-complete for instances where each edge has cost one or two. Moreover, the problem is W[1]-hard parameterized by k for instances with integer edge costs upper bounded by a polynomial of the size of the input graph.*

While Min s-t-Cut$-k$ exp is NP-complete on general graphs, our next theorem establishes that on graphs of bounded genus, *all* eight variants of the discounted cut problem are solvable in polynomial time when the maximum edge cost is polynomial in n. The proof relies on a generic approach that reduces discounted cuts with polynomial weights to their classic counterparts on bounded-genus graphs. This reduction builds on an algebraic technique [9] for handling generating functions of cuts in bounded-genus graphs.

Theorem 2. *Each of the eight problems*

- Min/MaxCut-k cheap/exp,
- Min/Max s-t-Cut-k cheap/exp

restricted to graphs of genus g, given with the corresponding embedding, admits a randomized algorithm of running time

$$\left(4^g \cdot n^{1.5} + m \cdot M\right) \cdot m^2 \cdot M \cdot \log^{\mathcal{O}(1)}(n + M),$$

where M is the total edge cost. The algorithm can be derandomized for the cost of additional multiplier of order n in the running time.

[1] Min s-t-Cut$-k$ exp is also known as Most Vital Links.

Theorem 3. *If* Minimum Π *(*Maximum Π*, respectively) can be solved in time* $T(|U|, \|\mathcal{F}\|, M)$ *then* Minimum Π with k Free Cheap Elements *(*Maximum Π with k Free Expensive Elements*, respectively) can be solved in time* $\mathcal{O}(|U| \cdot T(|U|, \|\mathcal{F}\|, M))$ *where* M *is the maximum value of the cost function.*

Hence, if Minimum Π is solvable in polynomial time, then Minimum Π with k Free Cheap Elements is as well, and if Maximum Π is, then so is Maximum Π with k Free Expensive Elements. By pipelining Theorem 3 with known complexity results for classic variants of cut problems, we obtain a number of corollaries.

Since Min s-t-Cut is polynomial-time solvable in almost linear time [6], by Theorem 3, Min s-t-Cut$-k$ cheap can be solved in $\mathcal{O}(m^{2+o(1)} \log M)$ time for instances with cost functions of maximum value M. Furthermore, this result holds for directed graphs.

Similarly, MinCut can be solved in polynomial time by the randomized Karger's algorithm [14] or the deterministic Stoer–Wagner algorithm [22]. In particular, using the improved version of Karger's algorithm given by Gawrychowski et al. [11], we have that MinCut$-k$ cheap is solvable in $\mathcal{O}(m^2 \log^2 n \log M)$ time by a randomized algorithm for instances with cost functions of maximum value M.

The classic MaxCut is NP-complete. However, MaxCut can be solved in polynomial time on planar graphs [12], graphs excluding K_5 as a minor [4], and graphs of bounded genus [9]. Combining Theorem 3 and the results from [16], we obtain that MaxCut$-k$ exp can be solved in $\mathcal{O}(n^{5/2} \log n \log M)$ time on planar graphs for instances with cost functions of maximum value M.

By Theorem 2, Min s-t-Cut$-k$ exp is solvable in polynomial time on graphs of bounded genus with polynomial costs. We show that on planar graphs, the constraint on costs can be excluded. Interestingly, already the first papers [23, 18] on Min s-t-Cut$-k$ exp studied planar graphs. However, polynomial time algorithms were known only for the situation when both terminal vertices s and t lie on outer face.

Theorem 4. *On planar graphs,* Min s-t-Cut$-k$ exp *can be solved in time* $\mathcal{O}(kn^2 \log n \log M)$ *where* M *is the maximum value of the cost function.*

The core idea in the proof of Theorem 4 is that computing a discounted cut in a planar graph G can elegantly be reduced to finding specific walks in its dual graph G^*. More specifically, every minimal cut in G corresponds to a cycle in G^*. Let us fix an s-t-path P in G. The key observation (sometimes known as the *discrete Jordan curve theorem*) is that a cycle in G^* corresponds to an s-t-cut in G if and only if it crosses P an odd number of times. Consequently, the problem reduces to finding a minimum-cost closed walk in G^* that intersects P an odd number of times, which can be solved using dynamic programming.

Finally, for MinCut$-k$ exp, we give a polynomial-time algorithm that works with arbitrary edge costs on general graphs. In light of the NP-hardness of Min s-t-Cut with k Free Expensive Edges (see Theorem 1), this result highlights a surprising difference in the complexity of these two problems.

Theorem 5. MinCut$-k$ exp *admits a randomized algorithm that runs in time* $\mathcal{O}(n^3 \cdot m \cdot \log^4 n \cdot \log\log n \cdot \log M)$ *where* M *is the maximum value of the cost function.*

The proof of Theorem 5 relies on an algorithm for the Bicriteria Global Minimum Cut. In this problem, we are given a graph G with two edge weight functions $w_1, w_2 \colon E(G) \to \mathbb{R}_{\geq 0}$ and real numbers $b_1, b_2 \geq 0$. The task is to decide whether there is a cut (A, B) of G such that $w_i(E(A, B)) \leq b_i$ for every $i \in \{1, 2\}$. By [2], this problem is solvable in polynomial time. (The algorithm of Armon and Zwick runs in polynomial time even in a more general setting for an arbitrary fixed number k of criteria. But for the proof of Theorem 5, we need only $k = 2$.) The fastest known algorithm for Bicriteria Global Minimum Cut is due to Aissi et al. [1], and provides a solution in time $\mathcal{O}(n^3 \log^4 n \log \log n)$ with probability $1 - 1/\Omega(n)$.

1.3 Related Work

The problem of determining the *most vital link* (singular) [23] in a graph is the following. Given a flow network (G, s, t, c), that is, a graph G with source and target vertices, and a capacity function, determine which edge in G should be removed to obtain a graph with the smallest possible maximum s-t-flow. Clearly, this can be done in polynomial time, since we can try every edge and run a max-flow algorithm. By the Max Flow–Min Cut theorem, we want to find an edge whose removal reduces the Min s-t-cut as much as possible. Wollmer studied the problem on s-t-planar graphs, i.e., where the flow network is planar and s and t are both on the outer face. Using a correspondence between minimal cuts and paths in planar graphs, Wollmer provided a linear-time algorithm for the problem on this class of graphs.

Wollmer extended the concept of most vital link to the Most Vital Links problem [24], where the objective is to remove k edges to minimize the s-t-flow. Again, by the Max Flow–Min Cut theorem, this is equivalent to deleting k edges to obtain a graph with as small a minimum s-t-cut as possible. Therefore, we want to find a minimal cut and delete edges only from that cut—k edges with highest capacities. The remaining edges will be the resulting cut. This implies that Most Vital Links is equivalent to Min s-t-Cut$-k$ exp.

Phillips [20] studied this variant and provided a Polynomial-Time Approximation Scheme (PTAS) for planar graphs. Later, Wood [25] claimed that the Most Vital Links on planar graphs had been solved. However, Wood's assertion merely referenced Wollmer's result, which applies exclusively to s-t-planar graphs. In fact, Wollmer's algorithm assumes s and t share a face in the planar embedding, as it starts by creating a direct edge between s and t, which is not feasible otherwise.

The Most Vital Links has been studied under various names across different fields. One such area is *graph vulnerability*, which examines how much a graph "breaks apart" when a certain number of edges or vertices are removed. Another is *network inhibition* [20] or *network interdiction* [25], where the goal is to "damage" a graph as effectively as possible while using minimal resources. McMasters and Mustin [18] study the problem of allocating a limited number of aircraft to interdict an enemy's supply lines on a particular day using and give a linear programming algorithm for the case where the cost of destroying an edge is the same as its capacity. Like Wollmer, they consider only s-t-planar graphs.

Later, several variations of the Most Vital Links problem were studied, mostly under the names Network Interdiction and Network Inhibition. Many of these problems were modeled as Stackelberg games, in which an in-

terdictor attempts to damage the network while a defender tries to repair or reinforce it [21]. Zenklusen [26] studied several interdiction problems, including flow interdiction on planar graphs and later, matching interdiction [27], i.e., removing edges to make the maximal matching as small as possible.

Interestingly, despite the Most Vital Links problem's long history and extensive bibliography, several open algorithmic questions remain about the minimum and maximum cuts whose cost is measured up to k edges. In this paper, we address these gaps and introduce additional algorithmic questions, hoping to stimulate further progress in this area.

2 Preliminaries

For a positive integer k, we write $[k]$ to denote the set $\{1, \ldots, k\}$.

Graphs. In this paper, we consider finite undirected graphs and refer to the textbook by Diestel [7] for undefined notions. We use $V(G)$ and $E(G)$ to denote the set of vertices and the set of edges of G, respectively. We use n and m to denote the number of vertices and edges in G, respectively, unless this creates confusion, in which case we clarify the notation explicitly.

For a vertex subset $U \subseteq V$, we use $G[U]$ to denote the subgraph of G induced by the vertices in U and $G - U$ to denote $G[V \setminus U]$. For a vertex v, we use $\deg_G(v)$ to denote its *degree*, i.e, the number of its neighbors; we may omit the index if G is clear from the context.

A *walk* $W = v_0 \ldots v_\ell$ of in a graph G is a sequence of (not necessarily distinct) vertices such that v_{i-1} and v_i are adjacent for all $i \in [\ell]$. We say that W is *odd* if ℓ is an odd integer. We also say that ℓ is the *length* of W for unweighted graphs. Given a cost/weight function $c\colon E(G) \to \mathbb{Z}_{\geq 0}$, the length of $W = v_0 \ldots v_\ell$ is defined as $\sum_{i=1}^{\ell} c(v_{i-1}v_i)$. We say that W is *closed* if $v_0 = v_\ell$. The vertices v_0 and v_ℓ are the *endpoints* of W and the other vertices are *internal*. A *path* is a walk with distinct vertices, and a *cycle* is a closed walk where all vertices except endpoint are distinct. Notice that a path $P = v_0 \ldots v_\ell$ is uniquely defined by its set of edges $E(P) = \{v_{i-1}v_i \mid i \in [\ell]\}$, and the same holds for a cycle. Thus, we may say that a set of edges is a path (cycle) if it is the set of edges of a path (a cycle, respectively). We also can consider a path or cycle to be a graph.

For two disjoint sets of vertices $A, B \subseteq V(G)$, we write $E(A, B) = \{uv \in E(G) \mid u \in A,\ v \in B\}$, An *(edge) cut* of G is a partition (A, B) of $V(G)$ and the set of edges $E(A, B)$ is the *cut-set*. We say that (A, B) is an s-t-cut for two vertices s and t if these vertices are in distinct sets of the partition. Given a cost function $c\colon E(G) \to \mathbb{Z}_{\geq 0}$, the *cost* of a cut (A, B) is defined as the sum of the cost of the edges of $E(A, B)$. A cut (A, B) is an *(inclusion) minimal* if there is no other cut (A', B') such that $E(A', B') \subset E(A, B)$. It is well-known that (A, B) is minimal if and only if $G[A]$ and $G[B]$ are connected.

Planar graphs and graphs on surfaces. A graph is *planar* if it can be drawn on the plane such that its edges do not cross each other. Such a drawing is called a *planar embedding* of the graph and a planar graph with a planar embedding is called a *plane* graph. The *faces* of a plane graph are the open regions of the plane bounded by a set of edges and that do not contain any other vertices or edges. We remind that a cycle of a plane graph has exactly two faces.

For a plane graph G, its *dual* graph $G^* = (F(G), E^*)$ has the set of faces of G as the vertex set, and for each $e \in E(G)$, G^* has the *dual* edge e^* whose endpoints are either two faces having e on their frontiers or e^* is a self-loop at f if e is in the frontier of exactly one face f (i.e., e is a bridge of G). Observe that G^* is not necessarily simple even if G is a simple graph. Throughout the paper, we will use $\circ^*$ to denote the *dual* of either a plane graph, a set of vertices/faces, or a set of edges, or a single element. Given a cost function $c\colon E(G) \to \mathbb{Z}_{\geq 0}$, we assume that $c(e^*) = c(e)$ for every $e \in E(G)$.

It is well known (see, e.g., [7]) that finding an inclusion minimal cut for a plane graph is equivalent to computing a cycle in the dual graph. More precisely, (A, B) is a minimal cut of a connected plane graph G if and only if $\{e^* \in E(G^*) \mid e \in E(A, B)\}$ is a cycle of G^*. Furthermore, (A, B) is a minimal s-t-cut if and only if $C = \{e^* \in E(G^*) \mid e \in E(A, B)\}$ is a cycle of G^* such that s and t are in distinct faces of C.

The genus of a graph is the minimal integer g such that the graph can be drawn without crossing itself on a sphere with g handles (i.e. an oriented surface of the genus g).

3 General Technique for Subset Problems with Free Elements

In this section, we present general results for minimization (maximization, respectively) problems with free cheap (expensive, respectively) elements. We show that, given an algorithm $\mathcal{A}$ for Minimum Π (Maximum Π, respectively), we can solve Minimum Π with k Free Cheap Elements (Maximum Π with k Free Expensive Elements, respectively) using at most $|U|$ calls of $\mathcal{A}$.

For an instance $(U, c, \mathcal{F})$ of Minimum Π (Maximum Π, respectively), let $\mathsf{Opt}_{\min}(U, c, \mathcal{F})$ ($\mathsf{Opt}_{\max}(U, c, \mathcal{F})$, respectively) be the optimum cost of a feasible solution, and we use $\mathsf{Opt}^{\mathrm{cheap}}_{\min}(U, c, \mathcal{F}, k)$ and $\mathsf{Opt}^{\mathrm{exp}}_{\max}(U, c, \mathcal{F}, k)$ for the optimum cost for Minimum Π with k Free Cheap Elements and Maximum Π with k Free Expensive Elements, respectively. We assume that these values are zeros if optimum subsets are of cardinality at most k. Let $c\colon U \to \mathbb{Z}_{\geq 0}$ be a function and let $w \geq 0$ be an integer. We use c_w and c^w for functions

$$c_w(x) = \begin{cases} c(x) & \text{if } x > w \\ w & \text{if } x \leq w \end{cases} \qquad c^w(x) = \begin{cases} c(x) & \text{if } x < w \\ w & \text{if } x \geq w \end{cases}$$

and write $C(X) = \{c(x) \mid x \in X\}$ for $X \subseteq U$.

Now, let

$$\mathrm{O}_{\min} = \min_{w \in C(U)} (\mathsf{Opt}_{\min}(U, c_w, \mathcal{F}) - kw)$$

and

$$\mathrm{O}_{\max} = \max_{w \in C(U)} (\mathsf{Opt}_{\max}(U, c^w, \mathcal{F}) - kw).$$

Lemma 1. *For an instance $(U, c, \mathcal{F})$ of* Minimum Π *(*Maximum Π*, respectively) and $k \geq 0$,*

- $\mathsf{Opt}^{\mathrm{cheap}}_{\min}(U, c, \mathcal{F}, k) = \max\{0, \mathrm{O}_{\min}\}$
- $\mathsf{Opt}^{\mathrm{exp}}_{\max}(U, c, \mathcal{F}, k) = \max\{0, \mathrm{O}_{\max}\}$.

Proof of Lemma 1. We prove the lemma for MINIMUM Π and MINIMUM Π WITH k FREE CHEAP ELEMENTS; the proof for the maximization variant uses symmetric arguments.

First, we prove that

$$\mathsf{Opt}^{\text{cheap}}_{\min}(U,c,\mathcal{F},k) \geq \max\{0, \min_{w\in C(U)}(\mathsf{Opt}_{\min}(U,c_w,\mathcal{F})-kw)\}.$$

Note that $\mathsf{Opt}^{\text{cheap}}_{\min}(U,c,\mathcal{F},k) \geq 0$ by definition. Thus, we have to show that $\mathsf{Opt}^{\text{cheap}}_{\min}(U,c,\mathcal{F},k) \geq \min_{w\in C(U)}(\mathsf{Opt}_{\min}(U,c_w,\mathcal{F})-kw)$. Assume that $S\subseteq U$ is a feasible set of optimum cost. We consider two cases.

Suppose that $|S|\leq k$. Then $\mathsf{Opt}^{\text{cheap}}_{\min}(U,c,\mathcal{F},k)=0$. Consider $w^*=\max C(U)$. Then $\mathsf{Opt}_{\min}(U,c_{w^*},\mathcal{F})\leq kw^*$ because S is a feasible solution of size at most k. Therefore, $\mathsf{Opt}_{\min}(U,c_{w^*},\mathcal{F})-kw^*\leq 0$ and $\mathsf{Opt}^{\text{cheap}}_{\min}(U,c,\mathcal{F},k)\geq \mathsf{Opt}_{\min}(U,c_{w^*},\mathcal{F})-kw^*\geq \min_{w\in C(U)}(\mathsf{Opt}_{\min}(U,c_w,\mathcal{F})-kw)$.

Assume that $|S|>k$ and let $X\subseteq S$ be the subset of k cheapest elements of S. We set $w^*=\max C(X)$. Then

$$\begin{aligned}\mathsf{Opt}^{\text{cheap}}_{\min}(U,c,\mathcal{F},k) =& c(S\setminus X)=c_{w^*}(S\setminus X)\\ =& c_{w*}(S)-c_{w^*}(X)=c_{w*}(S)-kw^*\\ &\geq \mathsf{Opt}_{\min}(U,c_{w^*},\mathcal{F})-kw^*\\ &\geq \min_{w\in C(U)}(\mathsf{Opt}_{\min}(U,c_w,\mathcal{F})-kw).\end{aligned}$$

For the opposite inequality

$$\mathsf{Opt}^{\text{cheap}}_{\min}(U,c,\mathcal{F},k) \leq \max\{0, \min_{w\in C(U)}(\mathsf{Opt}_{\min}(U,c_w,\mathcal{F})-kw)\},$$

assume first that there is a feasible set of size at most k. Then

$$\mathsf{Opt}^{\text{cheap}}_{\min}(U,c,\mathcal{F},k)=0$$

by definition and the inequality holds. Suppose that any feasible set is of size at most $k+1$. In particular, this means that $\max\{0,\min_{w\in c(U)}(\mathsf{Opt}_{\min}(U,c_w,\mathcal{F})-kw)=\min_{w\in C(U)}(\mathsf{Opt}_{\min}(U,c_w,\mathcal{F})-kw)$. Let $w^*\in C(U)$ and let S be a feasible set such that $\min_{w\in C(U)}(\mathsf{Opt}_{\min}(U,c_w,\mathcal{F})-kw)=\mathsf{Opt}_{\min}(U,c_{w^*},\mathcal{F})-kw^*=c_{w^*}(S)-kw^*$. Denote by $X\subseteq S$ the set of k cheapest elements of S with respect to c_{w^*}. By the definition of c_{w^*}, $c_{w^*}(x)\geq w^*$ for any $x\in U$. Then $c_{w^*}(S)-kw^*\geq c_{w^*}(S)-c_{w^*}(X)=c_{w^*}(S\setminus X)$. Since $c_{w^*}(x)\geq c_w(x)$ for all $x\in U$, $c_{w^*}(S\setminus X)\geq c(S\setminus X)\geq \mathsf{Opt}^{\text{cheap}}_{\min}(U,c,\mathcal{F},k)$. This proves the inequality and completes the proof of the lemma. □

Lemma 1 proves that MINIMUM Π WITH k FREE CHEAP ELEMENTS and MAXIMUM Π WITH k FREE EXPENSIVE ELEMENTS can be reduced to MINIMUM Π and MAXIMUM Π, respectively. This proves Theorem 3.

In particular, if MINIMUM Π or MAXIMUM Π can be solved in polynomial time then the corresponding problem with free edges admits a polynomial-time algorithm as well. It is well-known that MIN s-t-CUT is polynomial-time solvable and, by the recent results of Chen et al. [6], the minimum s-t-cut can be found in an almost linear time. This implies the following proposition.

Proposition 1. Min s-t-Cut$-k$ cheap *can be solved in* $\mathcal{O}(m^{2+o(1)} \log M)$ *time for instances with cost functions of maximum value* M*. Furthermore, this result holds for directed graphs.*

Similarly, MinCut can be solved in polynomial time by the randomized Karger's algorithm [14] or the deterministic Stoer–Wagner algorithm [22]. In particular, using the improved version of Karger's algorithm [11], we obtain the following proposition.

Proposition 2. MinCut$-k$ cheap *can be solved in* $\mathcal{O}(m^2 \log^2 n \log M)$ *time by a randomized algorithm for instances with cost functions of maximum value* M*.*

Finally in this section, we note that MaxCut can be solved in polynomial time on planar graphs [12], graphs excluding K_5 as a minor [4], and graphs of bounded genus [9]. We discuss this problem in detail in the next section. Here, we only mention that combining Theorem 3 and the results of Liers and Pardella [16], we obtain the following result for planar graphs.

Proposition 3. MaxCut$-k$ exp *can be solved in* $\mathcal{O}(n^{5/2} \log n \log M)$ *time on planar graphs for instances with cost functions of maximum value* M*.*

3.1 Lower Bounds for Min s-t-Cut with k Free Expensive Edges

In this section we prove that Min s-t-Cut with k Free Cheap Edges is NP-complete and W-hard even in very restricted cases. Notice that the problem can be solved in polynomial time when the edges of the input graph have unit costs—to obtain an optimum solution, it is sufficient to find the minimum s-t-cut in the considered graph and subtract k from its size. We prove that the problem becomes NP-hard when we allow edges to have cost 1 or 2 and show that Min s-t-Cut$-k$ exp is W[1]-hard when parameterized by k, the number of edges to delete, when we allow polynomial (in the graph size) costs.

Theorem 1. Min s-t-Cut$-k$ exp *is NP-complete for instances where each edge has cost one or two. Moreover, the problem is W[1]-hard parameterized by* k *for instances with integer edge costs upper bounded by a polynomial of the size of the input graph.*

Proof. First, we show NP-hardness and then explain how to modify the proof for the second claim. We reduce from the Clique problem which asks, given a graph G and a positive integer k, whether G contains a clique of size k. This problem is well-known to be NP-complete [10] and W[1]-complete when parameterized by k [8]. Furthermore (see [17]), it is known that these computational lower bounds hold for regular graphs. Notice that it is sufficient to prove Theorem 1 for multigraphs; to show the results for simple graphs, it is sufficient to subdivide each edge once and assign to the obtained two edges the same cost as the cost of the original edge.

Let (G, k) be an instance of the Clique problem where G is a d-regular graph. We set $p = (d-k+1)k$ and $q = 2p+2$. Then we construct the multigraph G' with edge costs as follows.

(i) Construct a copy of G, and assign cost one to each edge of G.

(ii) Add a vertex s, and for each $v \in V(G)$, add q edges sv of cost two.

(iii) Add a vertex t, and for each $v \in V(G)$, add $p+q+1$ edges tv of cost one.

We use c to denote the cost function for the obtained instance of Min s-t-Cut$-k$ exp. We define $k' = kq$ and set $\beta = p + (n-k)(p+q+1)$. We claim that (G,k) is a yes-instance of Clique if and only if there is an s-t-cut (A,B) of G' with $\mathsf{Cost}^{\exp}_{k'}(A,B) \leq \beta$.

Suppose that G has a clique K of size k. We define $A = \{s\} \cup (V(G') \setminus K)$ and $B = V(G') \setminus A$. Since $s \in A$ and $t \in B$, we have that (A,B) is an s-t-cut. The k' most expensive edges in $E(A,B)$ are the edges sv for $v \in K$; each of them is of cost two. The set $R = \{sv \mid v \in K\}$ contains $qk = k'$ edges. Because G' is d-regular, $c(E(A,B)) = qk + (d-k+1)k + (p+q+1)(n-k) = c(R) + p + (p+q+1)(n-1) = c(R) + \beta$. Thus, $c(E(A,B)) - c(R) = \beta$. This means that $\mathsf{Cost}^{\exp}_{k'}(A,B) \leq \beta$.

For the opposite direction, assume that $\mathsf{Cost}^{\exp}_{k'}(A,B) \leq \beta$ for an s-t-cut (A,B) of G' assuming that $s \in A$ and $t \in B$. Let $X = V(G) \cap B$ and $Y = V(G) \cap B$. We show that X is a clique of G of size k.

First, we show that $|X| = k$. The proof is by contradiction. Let $r = |X|$ and consider two cases.

Suppose that $r > k$. Then k' most expensive edges are edges sv for $v \in X$. The total number of edges of this type in $E(A,B)$ is $qr > qk = k'$. Let $R \subseteq E(A,B)$ be a set of k' edges sv for $v \in X$. Then $E(A,B)$ contains $(r-k)q$ edges $sv \in E(A,B) \setminus R$ with $v \in X$ and the total cost of these edges is $2(r-k)q$. Also $E(A,B)$ contains $(n-r)(p+q+1)$ edges tv for $v \in Y$ of cost one. The total cost of these edges is $(n-r)(p+q+1)$. Hence,

$$\begin{aligned}
c(E(A,B)) - c(R) &\geq 2(r-k)q + (n-r)(p+q+1) \\
&= (r-k)(q+p+1) + (r-k)(q-p-1) \\
&\quad + (n-r)(p+q+1) \\
&= (r-k)(q-p-1) + (n-k)(p+q+1) \\
&\geq (q-p-1) + (n-k)(p+q+1) \\
&= p+1+(n-k)(p+q+1) > \beta
\end{aligned}$$

contradicting that $\mathsf{Cost}^{\exp}_{k'}(A,B) \leq \beta$. Thus, $r \leq k$.

Suppose that $r < k$. Then k' most expensive edges are rq edges sv for $v \in X$ of cost two and $k' - rq = (k-r)q$ edges of cost one. By the construction of G', $E(A,B)$ contains every edge tv for $v \in Y$ and the total cost of these edges is $(n-r)(p+q+1)$. Therefore,

$$\begin{aligned}
\mathsf{Cost}^{\exp}_{k'}(A,B) &\geq (n-r)(p+q+1) - (k-r)q \\
&= (n-r)(p+q+1) - (k-r)(p+q+1) \\
&\quad + (k-r)(p+1) \\
&\geq (n-k)(p+q+1) + p + 1 > \beta;
\end{aligned}$$

a contradiction. We conclude that $|X| = k$.

Now we show that X is a clique of G. Because $|X| = k$, the set of k' the most expensive edges of $E(A,B)$ is $R = \{sv \mid v \in X\}$. Also $E(A,B)$ contains $(n-k)(p+q+1)$ edges tv for $v \in Y$. Because these edges are of cost one,

this implies that $c(E(A,B)) - c(R) = |E(X,Y)| + (n-k)(p+q+1) \leq \beta = p + (n-k)(p+q+1)$. Thus, $|E(X,Y)| \leq p = (d-k+1)k$. Because G is a d-regular graph, we obtain that $G[X]$ contains at least $\frac{1}{2}k(k-1)$ edges. Since $|X| = k$, we have that X is a clique of size k. This concludes the NP-hardness proof for Min s-t-Cut$-k$ exp.

To show that Min s-t-Cut$-k$ exp is W[1]-hard when parameterized by k, we modify the above construction of G' by replacing (ii) by (ii*):

(ii*) Add a vertex s, and for each $v \in V(G)$, add edge sv of cost q.

Then we use the same arguments as in the NP-hardness proof to show that (G,k) is a yes-instance of Clique if and only if there is an s-t-cut (A,B) of G' with $\mathsf{Cost}_k^{\exp}(A,B) \leq \beta$. This concludes the proof. □

4 Graphs of Bounded Genus

In this section, we prove that all eight variants of the *cuts with k free edges* problems can be solved in polynomial time in graphs of bounded genus, when restricted to polynomially-bounded integer edge costs. That is, our algorithms depend on M polynomially, where M is the total edge cost. We remark that our algorithms require an embedding of G into an orientable surface of genus g given as an input; such embeddings are known to be computable in time $2^{g^{\mathcal{O}(1)}}n$ [19, 15].

We rely heavily on the work of Galluccio et al. [9]. They show that integral total cut costs can be computed efficiently—through polynomial evaluation and interpolation—once the corresponding embedding is known. We outline their approach in the proof of the following lemma.

Lemma 2. *There is a randomized algorithm that, given an n-vertex graph G with positive integral edge costs, along with its embedding ϕ in an orientable surface of genus g, in time $(4^g \cdot M \cdot n^{1.5} + M^2) \cdot \log^{\mathcal{O}(1)}(n+M)$, where M is the total edge cost, computes a set $C \subseteq \{0,1,\ldots,M\}$, that satisfies*

- *if $w \in C$, then there is a cut of cost w in G;*
- *if there is a cut of cost w in G, then $w \in C$ with probability at least $\frac{1}{2}$.*

The algorithm can be derandomized for the cost of additional multiplier of order n in the running time.

Proof. Consider a polynomial

$$\mathcal{C}(G,x) = \sum_{\substack{A \cup B = V(G) \\ A \cap B = \emptyset}} x^{c(E(A,B))} = \sum_{w=0}^{M} a_w x^w.$$

Clearly, G has a cut of cost w if and only if the coefficient a_w of x^w in $\mathcal{C}(G,x)$ is non-zero.

Galluccio et al. [9] showed that for any prime p and integer $i \in \{0,\ldots,p-1\}$, the value $\mathcal{C}(G,i)$ modulo p can be computed in time $4^g \cdot n^{1.5} \cdot \log^{\mathcal{O}(1)}(n+M+p)$ if an embedding ϕ of G into a surface of genus g is given.

Algorithm 1: Algorithm resolving an instance (G, c, k, β) of a problem Π. If Π is s-t-Cut, s and t are also a part of the input. Additionally, an embedding ϕ of G into an oriented surface of genus g is given.

$e_1, e_2, \ldots, e_m \leftarrow$ ordering of $E(G)$ s.t. $c(e_i) \leq c(e_{i+1})$
$M \leftarrow 1 + \sum_{i=1}^{m} c(e_i)$
for $t \in [m+1]$ **do**
  $G', \phi' \leftarrow$ copy of G, ϕ
  for $i \in [m]$ **do**
    if $i < t$ **and** Π is Free Cheap **or**
    if $i \geq t$ **and** Π is Free Expensive **then**
      $c'(e_i) \leftarrow M$
    else
      $c'(e_i) \leftarrow c(e_i)$
    end if
    if Π is s-t-Cut **then**
      add edge e_{m+1} between s and t in G'
      attach a new handle between s and t in ϕ' and embed e_{m+1} in it in ϕ'
      $c'(e_{m+1}) \leftarrow m \cdot M + 1$
      $T \leftarrow c'(e_{m+1})$
    else
      $T \leftarrow 0$
    end if
  end for
  $C \leftarrow$ output of the algorithm of Lemma 2 applied to G', c' and ϕ'
  for $w' \in \{w - T : w \in C, w \geq T\}$ **do**
    $\beta' \leftarrow w' \pmod M$ {reimbursed cost}
    $k' \leftarrow \lfloor \frac{w'}{M} \rfloor$ {number of reimbursed edges}
    if $\beta' \leq \beta$ **and** $k' \leq k$ **and** Π is Min **or**
    if $\beta' \geq \beta$ **and** $k' \geq k$ **and** Π is Max **then**
      return Yes
    end if
  end for
end for
if $\beta = 0$ and Π is Max **then**
  return Yes
end if
return No

We now describe our algorithm. First, it computes a sequence of pairwise-distinct prime numbers $p_1, p_2, \ldots, p_{2n}$, each greater than M. This is done by iterating a number q starting from $M+1$, adding q to the sequence if it is prime, increase q and repeat until the sequence has exactly $2n$ primes. Then the algorithm picks $j \in [2n]$ uniformly at random and puts $p := p_j$. Next, the algorithm selects an index $j \in [2n]$ uniformly at random and sets $p := p_j$. For every $i \in \{0, 1, \ldots, M\}$, it calls the Galluccio–Loebl–Vondrák subroutine to evaluate $c_{p,i} = \mathcal{C}(G, i) \bmod p$. Using Lagrange interpolation with arithmetic modulo p, it then recovers the coefficients $a_{p,0}, a_{p,1}, \ldots, a_{p,M}$. Finally, it outputs the set $C_p := \{\, w \mid a_{p,w} \neq 0 \,\}$.

We argue that the algorithm is correct. Since $a_{p,w} \equiv a_w \pmod p$, C_p can only contain numbers corresponding to non-zero coefficients in $\mathcal{C}(G, x)$ that correspond to existent cut costs. Consider $w \in \{0, \ldots, M\}$ such that G admits a cut of total cost w. To see that $w \in C_p$ with probability at least $\frac{1}{2}$, note that $w \notin C_p$ if and only if p divides a_w. The number of cuts is bounded by 2^n, so $a_w \leq 2^n$, and at most n primes can divide a_w, while the algorithm choice is made uniformly at random from a set of $2n$ distinct primes.

We now explain the running time bound of the algorithm. The first part of the algorithm evaluates primes $p_1, p_2, \ldots, p_{2n}$. By properties of the prime number distribution, p_{2n} is bounded by $\mathcal{O}((M+n) \cdot \log(M+n))$. Since a primality test can be performed in polylogarithmic time, the time required to construct the sequence is $\mathcal{O}(p_{2n} \cdot \log^{\mathcal{O}(1)}(p_{2n}))$.

Next part of the algorithm evaluates $\mathcal{C}(G, x)$ in $M+1$ distinct values of x modulo p. Each evaluation is a call to the Gallucio-Loebl-Vondrák algorithm. The total running time of this part is bounded by $M \cdot 4^g \cdot n^{1.5} \cdot \log^{\mathcal{O}(1)}(n+M)$, since $\log p = \mathcal{O}(\log(M+n))$.

Third subroutine of the algorithm corresponds to the construction of a Lagrangian polynomial of degree M in $\mathrm{GF}(p)$. This is done in $\mathcal{O}(M^2)$ arithmetic operations modulo p. That is, in $M^2 \cdot \log^{\mathcal{O}(1)}(M+n)$ running time.

To see the upper bound from the lemma statement, sum up the upper bounds for second and third parts. Upper bound for the first part is dominated by the upper bound for the second part.

To derandomize the algorithm, one just needs to run the algorithm $n+1$ times for every choice of p among $p_1, p_2, \ldots, p_{n+1}$, and report the union of all obtained solutions C_p. □

The main result of this section is the following, restating from the introduction.

Theorem 2. *Each of the eight problems*

- Min/MaxCut-k cheap/exp,
- Min/Max s-t-Cut-k cheap/exp

restricted to graphs of genus g, given with the corresponding embedding, admits a randomized algorithm of running time

$$\left(4^g \cdot n^{1.5} + m \cdot M\right) \cdot m^2 \cdot M \cdot \log^{\mathcal{O}(1)}(n+M),$$

where M is the total edge cost. The algorithm can be derandomized for the cost of additional multiplier of order n in the running time.

Proof. We show that each of the eight problems can be solved via $m+1$ calls to the algorithm of Lemma 2.

Consider any ordering $e_1, e_2, \ldots, e_m$ of edges of G such that $c(e_i) \leq c(e_{i+1})$ for every $i \in [m-1]$. For a fixed threshold $t \in [m+1]$, we say that e_i is *cheap* if $i < t$ and *expensive* otherwise. For each of the eight problems, there is at least one correct choice of t, such that in an optimal solution cheap edges and expensive edges agree with the choice of t. That is, for Free Cheap problems, there is an optimal cut that contains at most k edges e_i with $i < t$, and reimbursed cost of this cut equals the sum of $c(e_i)$ for $i \geq t$ taken over edges that it contains. For Free Expensive problems, edges with $i \geq t$ are reimbursed, while costs of edges with $i < t$ are summed up in the final reimbursed cost.

Building on this idea, we present in Algorithm 1 a universal algorithm that depends only slightly on the specifics of the problem Π. Given an instance (G, c, k, β) of Π and embedding ϕ of G into an orientable surface, the algorithm first finds an ordering of edges as discussed above. Then, for each choice of $t \in [m+1]$, the algorithms constructs a new graph G' with a new cost function c'. Initially, G' is a copy of G and ϕ' is a copy of ϕ, while $c'(e_i) = c(e_i)$ for edges that should not be reimbursed with respect to t and $c'(e_i) = M$ for edges that should be reimbursed with respect to t. Depending on Π, these are either edges of form e_i for $i < t$ or edges of form e_i for $i \geq t$. Note that M is slightly increased in a way that M is strictly greater than the total edge cost.

Then, if Π is a s-t-Cut type of problem, the algorithm adds an extra edge e_{m+1} between s and t in G'. To extend ϕ' with e_{m+1}, the algorithm attaches a new handle between s and t and embeds e_{m+1} in this handle in ϕ'. This operation increases the surface genus by at most 1, so ϕ' is an embedding of G' in a surface of genus $g+1$. Then e_{m+1} receives cost $c'(e_{m+1}) = T$ for $T := m \cdot M + 1$. If Π is a global cut type of problem, then G', ϕ' remains the same, while the algorithm puts $T := 0$.

As its next step, the algorithm runs the algorithm of Lemma 2 as a subroutine applied to G' and c'. A result of this run is a set C of costs of some cuts of (G', c'). Then the algorithm looks for a total cost of form $T + k' \cdot M + \beta'$ in C, for $k' \geq 0$ and $\beta' \leq M$. If $k' \leq k$ and $\beta' \leq \beta$ (for Π being a Min type of problem) or $k' \geq k$ and $\beta' \geq \beta$ (for Π being a Max type of problem), the algorithm reports that the given instance is a yes-instance of Π and stops.

If the algorithm did not report a yes-instance for every choice of $t \in [m+1]$, then it finally reports that the given instance is a no-instance, with one exception: in case when $\beta = 0$ and Π is a Max type of problem, the algorithm reports a yes-instance. The description of the algorithm is finished.

We claim that the algorithm recognizes yes-instances with high probability.

Claim 1. *If (G, c, k, β) is a yes-instance of Π, then the algorithm returns* Yes *with probability at least $\frac{1}{2}$.*

Proof. First, we show the claim for Π being a Min type and Free Cheap type of problem. To see this, let (A, B) be a cut that certifies that the instance is a yes-instance. Let S be the set of edges of this cut in G and let $e_{j_1}, e_{j_2}, \ldots, e_{j_q}$ be the edges of S in an order corresponding to the ordering found by the algorithm. Consider $t = j_{\min\{q,k\}} + 1$. The reimbursed cost of (A, B) equals $w = \sum_{e_i \in S, i \geq t} c(e_i) \leq \beta < M$. Consider the cost of (A, B) in G' constructed for this choice of t. Let S' be the set of edges of (A, B) in G'. If Π is s-t-Cut, then

$e_{m+1} \in S'$, since (A, B) is a cut between s and t. Since, $S' \setminus \{e_{m+1}\} = S$, we have that the cost of $S' \setminus \{e_{m+1}\}$ is comprised of $\min\{q, k\}$ reimbursed edges that have cost M in (G', c'), and other edges that have the same costs as in (G, c) and constitute the reimbursed cost of S. Hence, the total cost of (A, B) in (G', c') equals $w' = T + \min\{q, k\} \cdot M + w$. With probability at least $\frac{1}{2}$, the algorithm of Algorithm 1 puts w' in its resulting set C. Since $\min\{q, k\} \leq k$ and $w \leq \beta$, the algorithm correctly reports a yes-instance and terminates when it encounters w' in C.

When Π is a MAX type of problem, instances with $\beta = 0$ are handled by the algorithm separately in its final step. We only have to consider instances with $\beta > 0$. Then a solution cut (A, B) always contains at least $k+1$ edges. Similarly to above, the cost of S' in (G', c') equals $w' = T + \min\{q, k\} \cdot M + w$, which now becomes $w' = T + k \cdot M + w$. The algorithm will encounter such w' in C with probability at least $\frac{1}{2}$ and report a yes-instance.

For Π being a FREE EXPENSIVE type of problem the proof is symmetrical. □

While we state Theorem 2 for all eight variants of discounted cuts, some of these problems admit polynomial-time algorithms on general graphs. The first observation is that the variants of "min $-k$ cheap" and "max $-k$ expensive" are not significantly different from their classic counterparts. Furthermore, this type of statement holds for optimization problems in the most general sense.

Consider the MINIMUM Π problem, whose task is, given a universe U, a cost function $c\colon U \to \mathbb{Z}_{\geq 0}$, and a family $\mathcal{F}$ of feasible subsets of U (which may be given implicitly), to find the minimum cost of a feasible subset. We define MINIMUM Π WITH k FREE CHEAP ELEMENTS as the problem whose input additionally includes an integer $k \geq 0$, and whose task is to find the minimum cost of a feasible subset, excluding the cost of the k cheapest elements. Similarly, we define MAXIMUM Π WITH k FREE EXPENSIVE ELEMENTS for the MAXIMUM Π problem, where the objective is to find the maximum cost of a feasible subset while excluding the cost of the k most expensive elements.

While Theorem 2 claims that eight problems admit polynomial-time algorithms, the algorithms are very similar and follow a common scheme. In the proof below, we present a general algorithm that adjusts its constructions depending on a specific problem Π.

Other than that, we have to show that the algorithm never returns YES on no-instances. Assume that the algorithm returned YES in line 25 (the other yes-instance-return is trivially correct). By Lemma 2, there is a cut (A, B) in G' of total cost $w' = T + k' \cdot M + \beta'$. If Π is s-t-CUT, then (A, B) is an s-t-cut since $w' \geq T$. Also, (A, B) contains exactly k' edges of form e_i for $i < t$ or $i \geq t$ depending on the reimbursement type of Π; the total cost of other edges equals β'. If Π is MIN type, then the total cost of (A, B) without $k' \leq k$ reimbursed edges in (G, c) is exactly β', hence with k reimbursed edges the cost of (A, B) can't become greater than $\beta' \leq \beta$. Symmetrically, if Π is MAX type, then the total cost of (A, B) is β', if $k' \geq k$ edges are reimbursed, an with k edges it can't become smaller. In either case, (A, B) certifies that (G, c, k, β) is a yes-instance of Π. The correctness of the algorithm follows.

The running time required to find the ordering and construct G', c' and ϕ' is dominated by the running time of the algorithm of Lemma 2. Since the maximum cost in c' is now bounded by $\mathcal{O}(m \cdot M)$, G' is embedded in a surface

of genus $g+1$, one call to the subroutine takes

$$4^{g+1} \cdot n^{1.5} \cdot m \cdot M + m^2 \cdot M^2$$

running time up to a factor polylogarithmic in $n+M$. Our algorithm makes at most $m+1$ calls to the subroutine before it terminates. Hence, its total running time is upper-bounded by

$$\left(4^g \cdot n^{1.5} + m \cdot M\right) \cdot m^2 \cdot M \cdot \log^{\mathcal{O}(1)}(n+M).$$

Note that the derandomization of the algorithm of Lemma 2 also derandomizes our algorithm for the cost of an additional multiplier of order n. This concludes the proof of Theorem 2. □

While Theorem 2 asserts that the eight problems admit polynomial-time algorithms, these algorithms share similarities and adhere to a common framework.

4.1 Planar Min s-t-Cut with k Free Expensive Edges

Now we prove Theorem 4 which shows that Min s-t-Cut$-k$ exp is solvable in polynomial time on planar graphs with exponential edge costs. Throughout the section, we assume that all graphs considered are plane, meaning they come equipped with an embedding as the planarity can be tested and a planar embedding can be found (if it exists) in linear time, as shown by Hopcroft and Tarjan [13].

The crucial idea behind our algorithm is to switch to the dual problem. Recall that (A,B) is a minimal s-t-cut in a connected graph G if and only if $C=\{e^* \in E(G^*) \mid e \in E(A,B)\}$ is a cycle of G^* such that s and t are in distinct faces of C. Observe that given an instance (G,c,s,t,k,β) of Min s-t-Cut$-k$ exp, an s-t-cut (A,B) of minimum discounted cost $\mathsf{Cost}^{\exp}_k(A,B)$ should be minimal. Thus, the problem reduces to finding a cycle C in G^* of minimum discounted cost such that s and t are in distinct faces of C. Our first task is to establish a criteria that guarantees that s and t are in distinct faces of a cycle of G^*.

Let G be a plane graph. We say that a path P in G *crosses* a cycle C^* of G^* in $e \in E(P)$ if C^* contains the edge $e^* \in E(G^*)$ that is dual to e. The *number of crosses* of P and C^* is the number of edges of P where P and C^* cross. We can make the following observation.

Observation 1 (Bentert et al. [5])**.** *Let G be a plane graph, let $s,t \in V(G)$, and let P be an s-t-path. For any cycle C^* of G^*, s and t are in distinct faces of C^* if and only if the number of crosses of P and C^* is odd.*

Thus, to solve Min s-t-Cut$-k$ exp for (G,c,s,t,k,β), we have to find a cycle C in G^* of minimum discounted cost with odd number of crosses with some s-t-path in G. To solve this problem, we use the polynomial Dijkstra-style algorithm finding an odd s-t-walk of minimum discounted cost in a general graph. Let G be a graph with a given cost function $c\colon E(G) \to \mathbb{Z}_{\geq 0}$. Let also $k \geq 0$ be an integer. For a walk $W = v_0 \dots v_\ell$, let R be the sequence of k most expensive edges $v_{i-1}v_i$ in W; note that R may include several copies of the same edge of G. Then we define $\mathsf{Cost}^{\exp}_k(W) = \sum_{i=1}^{\ell} c(v_{i-1}v_i) - \sum_{e \in R} c(e)$.

We note that the following lemma holds for all graphs:

Lemma 3. *Given a graph G with a cost function $c\colon E(G) \to \mathbb{Z}_{\geq 0}$, two vertices s and t, and an integer $k \geq 0$, an odd s-t-walk W with minimum $\mathsf{Cost}_k^{\exp}(W)$ can be found in $\mathcal{O}(mk \log n \log M)$ time where M is the maximum value of the cost function.*

Proof Lemma 3. We present a Dijkstra-style dynamic programming algorithm (Algorithm 2). The DP table $\mathrm{dist}(k', p, v) = \ell$ records the minimum length ℓ of a walk from s to v that uses k' free edges and has parity $p \in \{0, 1\}$.

We work with *vectors* of the form (ℓ, k', p, v), which denotes that the shortest path from s to v using a number of edges congruent to p where k' edges are free, has cost ℓ.

Claim 2. *$\mathrm{dist}(k', p, v) = \ell$ is correct when the algorithm terminates. In addition, the first time $(\circ, k', p, v)$ is popped, its value is correct.*

Proof of claim. Notice that $(0, 0, 0, s)$ is the first vector to be popped, and is correct. Suppose that (ℓ, k, p, v) is the first time any $(\circ, k, p, v)$ vector is popped and is incorrect, and furthermore, is the first such incorrect vector to be popped.

Then there either (a) exists a walk from s to v using k free edges with parity p with length less than ℓ, or (b) there is no such s-v-walk with length ℓ.

In either case, (ℓ, k, p, v) was necessarily pushed onto the queue by a neighbor $u \in N(v)$ whose active vector at the time was $(\ell', k'\overline{p}, u)$, which has either (i) $(\ell - w(u, v), k, \overline{p}, u)$, or (ii) $(\ell, k - 1, \overline{p}, u)$. Since $(\ell', k', \overline{p}, u)$ was popped prior to (ℓ, k, p, v), it is correct by assumption, hence (ℓ, k, p, v) is correct as well, contradicting the assumption.

It follows that the first (and only) time we write (k, p, v) to the dist map, the value is correctly written. □

We now analyze the running time of Algorithm 2, aiming to show it is bounded by $\mathcal{O}(mk \log n \log M)$. If we can bound the number of iterations the of the while loop to t, we are done, since the work done inside the loop then is bounded by $\mathcal{O}(\deg(v) \cdot \log t)$ where v is the vertex popped on Line 5.

Claim 3. *v is pushed at most $2k \deg(v)$ many times.*

Proof. When a neighbor $u \in N(v)$ pushes v onto the queue, u is popped off, and by the previous claim, us vector is correct for its current k value, and hence will not push v more than that time. When it pushes v onto the queue, it pushes it at most twice, one with $k' = k$ and one with $k' = k + 1$, hence v is pushed at most $2k \deg(v)$ many times. □

From the claim above, each vertex is popped at most $4k \deg(v)$ many times, hence each vertex is pushed at most $4k \deg(v)$ times. In total $8km = \mathcal{O}(mk)$ many push and pop operations are performed, each running in time $\mathcal{O}(\log mk) = \mathcal{O}(\log n)$. Since the remaining computations are all $\mathcal{O}(\log M)$, the total running time is $\mathcal{O}(mk \log n \log M)$. □

We also use the following folklore observation.

Observation 2. *If $G = (V, E)$ contains an odd closed walk $W = v_0 \dots v_\ell$, then it also contains an odd cycle C that is part of W, that is, $V(C) \subseteq \{v_0, \dots, v_\ell\}$ and for every $e \in E(C)$, there is $i \in [\ell]$ such that $e = v_{i-1}v_i$.*

Proof. Let W be a closed odd walk and consider the graph B with $V(B) = \{v_0, \ldots, v_\ell\}$ and $E(B) = \{e \in E(G) \mid e = v_{i-1}v_i \text{ for some } i \in [\ell]\}$, i.e., the graph containing the vertices and all edges in the walk. Since W is a closed odd walk, B is not bipartite (since there are no closed odd walks in a bipartite graph), hence B has an odd cycle C. This completes the proof. □

Algorithm 2: Cheapest odd walk with k free edges

```
1: Create a priority queue Q
2: Q.push(0, 0, 0, s)
3: dist(v, k, p) ← ∞                          {all values initialized to ∞}
4: while Q ≠ ∅ do
5:    (ℓ, k', p, v) ← Q.pop()
6:    if dist(k', p, v) ≤ ℓ then
7:       continue                              {discard sub-optimal vector}
8:    end if
9:    dist(k', p, v) ← ℓ
10:   for u ∈ N_G(v) do
11:      Q.push(ℓ + w(u, v), k', p̄, u)
12:      if k' < k then
13:         Q.push(ℓ, k' + 1, p̄, u)
14:      end if
15:   end for
16: end while
```

Now we are ready to prove Theorem 4 which we restate:

Theorem 4. *On planar graphs,* Min s-t-Cut$-k$ exp *can be solved in time* $\mathcal{O}(kn^2 \log n \log M)$ *where* M *is the maximum value of the cost function.*

Proof. Let G be a plane graph with a cost function $c\colon E(G) \to \mathbb{Z}_{\geq 0}$, and let s and t be two distinct vertices, and $k \geq 0$ be an integer. We assume that G is connected as Min s-t-Cut with k Free Expensive Edges is trivial if s and t are in distinct connected components.

We find an arbitrary s-t-path P in G. Then we construct the dual graph G^* with the corresponding cost function c^*. For each edge $e^* = uv \in E(G^*)$ such that the dual edge $e \notin E(P)$, we subdivide this edge, that is, we introduce a new vertex w and replace uv with uw and wv. Denote the obtained graph $G^\star$. We define the cost function $c^\star\colon E(G^\star) \to \mathbb{Z}_{\geq 0}$ by setting $c^\star(e^*) = c^*(e^*) = c(e)$ for $e \in E(P)$, and we set $c^\star(uw) = c^*(uv)$ and $c^\star(wv) = 0$ if uw and wv were obtained by subdividing $e^* = uv$. The construction of $G^\star$ and the definition of the cost function $c^\star$ immediately imply the following property.

Claim 4. *The graph* G^* *has a cycle* C^* *with odd number of crossing of* P *if and only if* $G^\star$ *has an odd cycle* $C^\star$*. Moreover,* $\min \mathsf{Cost}_k^{\exp}(C^*)$ *taken over all cycles* C^* *with odd number of crossing of* P *is the same as* $\min \mathsf{Cost}_k^{\exp}(C^\star)$ *where the minimum is taken over all odd cycles* $C^\star$ *in* $G^\star$*.*

Thus, we reduced our initial task to finding an odd cycle in $G^\star$ of minimum discounted cost. To find such a cycle, we use the algorithm from Lemma 3. We go trough every vertex $x \in V(G^*)$, and run the algorithm to find an odd

x-x-walk W with minimum $\mathsf{Cost}_k^{\exp}(W)$. Among all such walks, we find a closed walk $W^\star$ of minimum discounted cost. By Observation 2, there is an odd cycle $C^\star$ that is a part of $W^\star$. Notice that $\mathsf{Cost}_k^{\exp}(C^\star) \leq \mathsf{Cost}_k^{\exp}(W^\star)$. Because $W^\star$ is an odd closed walk of minimum discounted cost, we obtain that $C^\star$ is an odd cycle in $G^\star$ of minimum discounted cost $\mathsf{Cost}_k^{\exp}(C^\star) = \mathsf{Cost}_k^{\exp}(W^\star)$. Since the minimum value of $\mathsf{Cost}_k^{\exp}(A, B)$ of an s-t-cut (A, B) of G is the same as $\mathsf{Cost}_k^{\exp}(C^\star)$, we return the cut corresponding to the cycle $C^\star$ in $G^\star$. This concludes the description of the algorithm and its correctness proof. It is easy to verify that the running is $\mathcal{O}(kn^2 \log n \log M)$.

□

5 Global MinCut with k Free Expensive Edges

This section is devoted to the *global* version of the minimum cut problem with reimbursed edges, namely MinCut$-k$ exp. The problem is well-known for the case $k = 0$, and the state-of-the-art randomized algorithm is due to Gawrychowski et al. [11] and runs in $\mathcal{O}(m \cdot \log^2 n \cdot \log M)$ time, where M is the maximum edge cost value.

The *multi-criteria* global minimum cut problem, where edges have multiple weight types that are measured separately, was introduced by Armon and Zwick [2] and later improved by Aissi et al. [1]. In this work, we focus on the two-criteria case and observe that MinCut$-k$ exp is a special case of the Bicriteria Global Minimum Cut problem: Given a graph with two edge weight functions w_1 and w_2, does there exist a cut (A, B) of G such that $w_i(E(A, B)) \leq b_i$ for every $i \in \{1, 2\}$?

Proposition 4 (Aissi et al. [1])**.** Bicriteria Global Minimum Cut *admits a randomized algorithm with $\mathcal{O}(n^3 \cdot \log^4 n \cdot \log \log n \cdot \log M)$ running time, where M is the maximum edge cost value.*

We use Proposition 4 to prove Theorem 5. The basic idea is to consider $m+1$ choices of splitting edges into *cheap* and *expensive* and construct an instance of Bicriteria Global Minimum Cut for each choice. Before giving the proof, we restate Theorem 5 for the reader's convenience.

Theorem 5. MinCut$-k$ exp *admits a randomized algorithm that runs in time $\mathcal{O}(n^3 \cdot m \cdot \log^4 n \cdot \log \log n \cdot \log M)$ where M is the maximum value of the cost function.*

Proof. We present an algorithm $\mathcal{A}$ that makes $m + 1$ calls to the algorithm of Proposition 4. Denote the given instance of MinCut$-k$ exp by (G, c, k, W). First, the algorithm finds an ordering $e_1, e_2, \ldots, e_m$ of edges of G such that their costs are non-decreasing: $c(e_1) \leq c(e_2) \leq \ldots \leq c(e_m)$. This is done via any sorting algorithm that runs in $\mathcal{O}(m \log m)$ time.

Then, the algorithm iterates t over values in $\{1, 2, \ldots, m, m + 1\}$. For a fixed choice of t, the algorithm constructs an instance $\mathcal{I}_t$ of Bicriteria Global Minimum Cut. The instance $\mathcal{I}_t$ is given by (G, w_1^t, w_2^t, W, k). The idea is that the first weight function w_1^t corresponds to the original edge costs of cheap edges given by c, while the second weight function w_2^t corresponds to the number of reimbursed expensive edges. For every $i < t$, the algorithm defines $w_1^t(e_i) := c(e_i)$, while in the same time $w_2^t(e_i) := 0$. These edges correspond to the edges that

are included in the cost. For every $i \geq t$, we define $w_2^t(e_i) := 1$, while $w_1^t(e_i)$ receives value zero. These edges correspond to the reimbursed edges.

The algorithm $\mathcal{A}$ passes each of the instances $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_{m+1}$ to the algorithm of Proposition 4. If at least one of these instances is a yes-instance, $\mathcal{A}$ reports that (G, c, k, W) is a yes-instance. Otherwise, $\mathcal{A}$ reports that it is a no-instance.

The running time upper bound for $\mathcal{A}$ is clear. We prove that $\mathcal{A}$ is correct. First, suppose (G, c, k, W) is a yes instance of MinCut$-k$ exp. Let (A, B) be the solution to (G, c, k, W). Let F be the subset of expensive edges of $E(A, B)$ that are free, while P is the set of edges that are included in the cost. Since $c(e_i) \leq c(e_j)$ for each $e_i \in P$, $e_j \in F$, we can assume $i < j$, i.e. the edge e_i is always reimbursed *after* the edge e_j for $i < j$. We have $|F| \leq k$ and $c(P) \leq W$ (note that $|F| < k$ only if $|P| = 0$). Let t be the smallest number in $\{i : e_i \in F\}$, or $m + 1$ if F is empty. Then the same cut (A, B) is a solution to $\mathcal{I}_t$:

$$w_1^t(P) + w_1^t(F) = w_1^t(P) + 0 = c(P) \leq W,$$

and

$$w_2^t(P) + w_2^t(F) = 0 + w_2^t(F) = |F| \leq k.$$

Towards opposite direction, let $\mathcal{I}_t$ be a yes-instance of Bicriteria Global Minimum Cut for some $t \in \{1, \ldots, m + 1\}$. Let (A, B) be its solution cut. Split $E(A, B)$ in F and P, F consists of all edges $e \in E(A, B)$ with $w_2^t(e) = 1$, and P consists of all other edges in $E(A, B)$. By definition of $\mathcal{I}_t$, $c(e) \leq c(e')$ for every $e \in P$, $e' \in F$. In the same time, $c(P) = w_1^t(E(A, B)) \leq W$ and $|F| \leq k$. Consequently, $\mathsf{Cost}_k^{\mathrm{exp}}(A, B) \leq W$ and (G, c, k, W) is a yes-instance of MinCut$-k$ exp. The proof is complete. □

6 Conclusion

We initiated a comprehensive study of cut problems with discounts and conclude with three open problems. We showed that on general graphs, Min s-t-Cut$-k$ exp is W[1]-hard parameterized by k for instances with integer edge costs upper bounded by a polynomial of the size of the input graph. Does the problem become fixed-parameter tractable FPT when the edge costs are constants? Is the problem FPT when parameterized by combined parameters k and the maximum edge cost? Our polynomial time algorithm for MaxCut$-k$ cheap on graphs of bounded genus relies on the assumption that the costs are polynomial. Is the problem polynomial time solvable on planar graphs for arbitrary weights?

References

[1] Hassene Aissi, Ali Ridha Mahjoub, and R. Ravi. Randomized contractions for multiobjective minimum cuts. In *25th Annual European Symposium on Algorithms (ESA)*, volume 87 of *LIPIcs*, pages 6:1–6:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017.

[2] Amitai Armon and Uri Zwick. Multicriteria Global Minimum Cuts. *Algorithmica*, 46(1):15–26, 2006. ISSN 1432-0541.

[3] Michael O. Ball, Bruce L. Golden, and Rakesh V. Vohra. Finding the most vital arcs in a network. *Operations Research Letters*, 8(2):73–76, 1989.

[4] Francisco Barahona. The max-cut problem on graphs not contractible to K_5. *Operations Research Letters*, 2(3):107–111, 1983. ISSN 0167-6377.

[5] Matthias Bentert, Pål Grønås Drange, Fedor V. Fomin, Petr A. Golovach, and Tuukka Korhonen. Two-sets cut-uncut on planar graphs. In *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia*, volume 297 of *LIPIcs*, pages 22:1–22:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024.

[6] Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Almost-linear-time algorithms for maximum flow and minimum-cost flow. *Commun. ACM*, 66(12):85–92, 2023.

[7] Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012. ISBN 978-3-642-14278-9.

[8] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. ISBN 978-1-4471-5558-4.

[9] Anna Galluccio, Martin Loebl, and Jan Vondrák. Optimization via enumeration: a new algorithm for the max cut problem. *Mathematical Programming*, 90:273–290, 2001.

[10] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979. ISBN 0-7167-1044-7.

[11] Pawel Gawrychowski, Shay Mozes, and Oren Weimann. Minimum cut in $O(m \log^2 n)$ time. *Theory Comput. Syst.*, 68(4):814–834, 2024.

[12] Frank Hadlock. Finding a maximum cut of a planar graph in polynomial time. *SIAM Journal on Computing*, 4(3):221–225, 1975.

[13] John E. Hopcroft and Robert Endre Tarjan. Efficient planarity testing. *J. ACM*, 21(4):549–568, 1974.

[14] David R. Karger and Clifford Stein. A new approach to the minimum cut problem. *Journal of the ACM*, 43(4):601–640, 1996. ISSN 0004-5411, 1557-735X.

[15] Ken-ichi Kawarabayashi, Bojan Mohar, and Bruce A. Reed. A simpler linear time algorithm for embedding graphs into an arbitrary surface and the genus of graphs of bounded tree-width. In *49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 771–780. IEEE Computer Society, 2008.

[16] Frauke Liers and G. Pardella. Partitioning planar graphs: A fast combinatorial approach for max-cut. *Comput. Optim. Appl.*, 51(1):323–344, 2012.

[17] Luke Mathieson and Stefan Szeider. Editing graphs to satisfy degree constraints: A parameterized approach. *J. Comput. Syst. Sci.*, 78(1):179–191, 2012.

[18] Alan W. McMasters and Thomas M. Mustin. Optimal interdiction of a supply network. *Naval Research Logistics Quarterly*, 17(3):261–268, 1970. ISSN 1931-9193.

[19] Bojan Mohar. A linear time algorithm for embedding graphs in an arbitrary surface. *SIAM Journal on Discrete Mathematics*, 12(1):6–26, 1999. ISSN 0895-4801.

[20] Cynthia A. Phillips. The network inhibition problem. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 776–785, 1993.

[21] Robert Louis Steinrauf. Network interdiction models. Master's thesis, Naval Postgraduate School Monterey Ca, Monterey, California, 1991.

[22] Mechthild Stoer and Frank Wagner. A simple min-cut algorithm. *Journal of the ACM (JACM)*, 44(4):585–591, 1997.

[23] Richard D. Wollmer. Some methods for determining the most vital link in a railway network. Technical report, RAND Corporation, 1963.

[24] Richard D. Wollmer. Removing arcs from a network. *Operations Research*, 12(6):934–940, 1964. ISSN 0030-364X.

[25] R. Kevin Wood. Deterministic network interdiction. *Mathematical and Computer Modelling*, 17(2):1–18, 1993. ISSN 0895-7177.

[26] Rico Zenklusen. Network flow interdiction on planar graphs. *Discrete Applied Mathematics*, 158(13):1441–1455, 2010. ISSN 0166218X.

[27] Rico Zenklusen. Connectivity interdiction. *Oper. Res. Lett.*, 42(6-7):450–454, 2014.